Date: 06/02/2025

What is a REST API?

REST (Representational State Transfer) is an architectural style for designing networked applications. REST APIs (Application Programming Interfaces) are interfaces that allow different software systems to communicate over HTTP (Hypertext Transfer Protocol).

Key Principles of REST:

1. **Statelessness**:
    - Each API call is independent and contains all the information required for processing.
    - The server does not store any client context between requests.
2. **Client-Server Architecture**:
    - The client and server are separate, allowing them to evolve independently.
    - The client is responsible for the user interface, while the server manages data storage and processing.
3. **Uniform Interface**:
    - Ensures consistency across APIs, simplifying interaction for developers.
    - Typically involves standardized URLs, HTTP methods (GET, POST, PUT, DELETE), and status codes.
4. **Resource-Based**:
    - Everything in a RESTful system is a **resource**, which can be data or functionality.
    - Resources are identified using URLs (Uniform Resource Locators).
5. **Stateless Communication**:
    - Communication is stateless, meaning each request from the client contains all the necessary information.
    - No session or user state is stored on the server between requests.
6. **Cacheable**:
    - Responses must define themselves as cacheable or non-cacheable to improve efficiency.
7. **Layered System**:
    - A client cannot tell whether it is connected directly to the server or through an intermediary like a load balancer or cache.

8. **Code on Demand (Optional)**:
    - Servers can temporarily extend or customize client functionality by transferring executable code (e.g., JavaScript).

Common HTTP Methods in REST:

1. **GET**:
    - Retrieves data from a server.
    - Example: GET /users returns a list of users.
2. **POST**:
    - Submits new data to the server.
    - Example: POST /users creates a new user.
3. **PUT**:
    - Updates existing data.
    - Example: PUT /users/1 updates the user with ID 1.
4. **DELETE**:
    - Deletes data from the server.
    - Example: DELETE /users/1 removes the user with ID 1.
5. **PATCH**:
    - Partially updates existing data.
    - Example: PATCH /users/1 updates some fields of user 1.

HTTP Status Codes:

- **200 OK**: Request succeeded.
- **201 Created**: Resource created successfully.
- **204 No Content**: Request succeeded, but no content to return.
- **400 Bad Request**: Invalid request format or parameters.
- **401 Unauthorized**: Authentication required or failed.
- **403 Forbidden**: Access is denied.
- **404 Not Found**: Resource not found.
- **500 Internal Server Error**: Server-side error.

Example of a RESTful API:

Assume we have a REST API for managing a library:

1. **GET /books**:
    - Retrieves a list of books.
2. **GET /books/5**:
    - Retrieves details of the book with ID 5.
3. **POST /books**:
    - Adds a new book to the library.
4. **PUT /books/5**:
    - Updates the details of the book with ID 5.
5. **DELETE /books/5**:
    - Deletes the book with ID 5.

Advantages of REST APIs:

1. **Simplicity**: Easy to use and understand due to standard HTTP methods and status codes.
2. **Scalability**: Statelessness and layered architecture promote scalability.
3. **Flexibility**: Supports various data formats like JSON, XML, and plain text.
4. **Performance**: Caching can improve response times for repeat requests.

Limitations of REST APIs:

1. **Statelessness**: Requires clients to handle more state, which can complicate interactions.
2. **Over-fetching/Under-fetching**: Can lead to inefficient data usage if not designed carefully.
3. **Limited Transactions**: Lacks support for complex transactions across multiple resources.

REST APIs are widely used due to their simplicity, scalability, and efficiency. They are the backbone of many web and mobile applications, enabling smooth communication between diverse systems.

Public APIs (Open APIs)

Public APIs are **openly available** for developers to use without any strict access restrictions. They are typically provided by organizations, services, or governments to allow third-party developers to integrate with their platforms. Some require API keys, while others are completely open.

Examples of Public APIs

Here are some widely used public APIs across different categories:

1. Weather APIs

- **OpenWeatherMap API**: Provides weather data, forecasts, and historical weather.
    - Docs: https://openweathermap.org/api
- **WeatherAPI**: Real-time and forecast weather data.
    - Docs: https://www.weatherapi.com/

2. Finance & Cryptocurrency APIs

- **CoinGecko API**: Provides cryptocurrency market data, including prices and trends.
    - Docs: https://www.coingecko.com/en/api
- **Alpha Vantage API**: Stock and cryptocurrency data.
    - Docs: https://www.alphavantage.co/
- **Open Exchange Rates API**: Currency exchange rates.
    - Docs: https://openexchangerates.org/